НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

з лабораторної роботи № 4

із дисципліни «Криптографічні методи захисту інформації»

на тему

*Шифрування з відкритим ключем на основі алгоритму RSA*

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-81 | *ст. викладач Бай Ю. П.* |
| *Верзун П.В.* |  |

Київ — 2020

ЗМІСТ

[Постановка задачі 2](#_Toc55342474)

[Основні теоретичні відомості з асиметричних криптосистем 3](#_Toc55342475)

[Математичне підґрунтя алгоритму RSA 3](#_Toc55342476)

[Контрольний приклад 1 4](#_Toc55342477)

[Контрольний приклад 2 5](#_Toc55342478)

[Шифрування і розшифрування за алгоритмом RSA 6](#_Toc55342479)

[Список літератури 7](#_Toc55342480)

[Додаток 1 8](#_Toc55342481)

[Додаток 2 9](#_Toc55342482)

***Мета роботи:*** розробити асиметричну криптосистему на основі алгоритму шифрування RSA.

# Постановка задачі

1. Скласти програму, яка дозволяє генерувати прості числа в заданому діапазоні та виконувати шифрування та розшифрування за алгоритмом RSA. Перевірити роботу програми на контрольних прикладах. Навести скріншоти детального покрокового виконання алгоритму.

1.*а*. Контрольний приклад 1 ([RSA-encryption](https://brilliant.org/wiki/rsa-encryption/))

*p* = 11, *q* = 17, *e* = 3

*public key* {*e, n*}= {3*,* 187}

*private key* {*d, n*}= {107*,* 187}

*M* = 72

*C* = 183

*M’* = 72, *text* = {H}

1.*б*. Контрольний приклад 2 ([RSA uk.wiki](https://uk.wikipedia.org/wiki/RSA))

*p* = 3557, *q* = 2579, *e* = 3

*public key* {*e, n*}= {3*,* 9173503}

*private key* {*d, n*}= {6111579*,* 9173503}

*M* = 1111111

*C* = 4051753

*M’* = 1111111

2. Виконати дії ОДЕРЖУВАЧА і розшифрувати задане повідомлення, користуючись алгоритмом RSA. Необхідні результати занести до [Таблиця RSA 3 курс](https://docs.google.com/spreadsheets/d/1gtZC9geHMUL4dCh347UmVJKLqED8NJ-CmLgI2NUZxDc/edit?usp=sharing).

УВАГА! Числа ***n*** в [Таблиця RSA 3 курс](https://docs.google.com/spreadsheets/d/1gtZC9geHMUL4dCh347UmVJKLqED8NJ-CmLgI2NUZxDc/edit?usp=sharing) мають бути унікальними.

# Основні теоретичні відомості з асиметричних криптосистем

Асиметричні криптосистеми — ефективні системи криптографічного захисту даних, які також називають криптосистемами з відкритим ключем. В таких системах для зашифровування даних використовують один ключ, а для розшифровування — інший (звідси і назва — асиметричні). Перший ключ є відкритим і може бути опублікованим для використання усіма користувачами системи, які шифрують дані. Розшифровування даних за допомогою відкритого ключа неможливе. Для розшифровування даних отримувач зашифрованої інформації використовує другий ключ, який є секретним (закритим). Зрозуміло, що ключ розшифровування не може бути визначеним з ключа зашифровування.

Головне досягнення асиметричного шифрування в тому, що воно дозволяє людям, що не мають наперед наявної домовленості про безпеку, обмінюватися секретними повідомленнями. Необхідність відправникові й одержувачеві погоджувати таємний ключ по спеціальному захищеному каналу цілком відпала. Прикладами криптосистем з відкритим ключем є Схема Ель-Гамаля (названа на честь автора, Тахера Ель-Гамаля), RSA (названа на честь винахідників: Рона Рівеста, Аді Шаміра і Леонарда Адлмана), Діффі-Геллмана і DSA, англ. Digital Signature Algorithm (винайдений Девідом Кравіцом).

# Опис алгоритму RSA

**Крок перший. підготовка ключів**

Спочатку необхідно виконати попередні дії: згенерувати публічний і приватний ключ:

* Обирати два простих числа p і q.
* Обчислити модуль добуток p і q: n = p × q.
* Обчислити функцію Ейлера: φ = (p-1) × (q-1).
* Підбірати число e, що відповідає таким критеріям:
  + вона повинна бути просте,
  + воно повинно бути менше φ
  + воно повинно бути взаємно просте з φ

Тепер пара чисел {e, n} - це відкритий ключ. Він відправляється тому, хто має шифрувати своє повідомлення. Далі необхідно розрахувати закритий ключ.

Для цього треба обчислити число d, зворотне е по модулю φ. Тобто залишок від ділення по модулю φ добутку d × e має дорівнювати 1. Запишемо це в позначеннях, прийнятих в багатьох мовах програмування: (d × е)% φ = 1. Пара {d, n} - це секретний ключ. Його не можна повідомляти нікому. Тільки володар секретного ключа може розшифрувати те, що було зашифровано відкритим ключем.

**Крок другий. шифрування**

Шифрування виконується за наступним алгоритмом:

* Звести повідомлення в ступінь e по модулю n.

Тут треба зауважити, що повідомлення P не повинно бути більше n. інакше нічого не вийде.

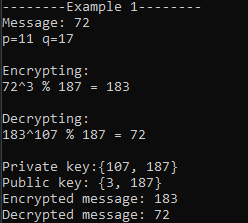
**Крок третій. розшифровка**

Важливо відмітити, що відкритий ключ не може розшифрувати повідомлення.

Починаємо розкодовувати:

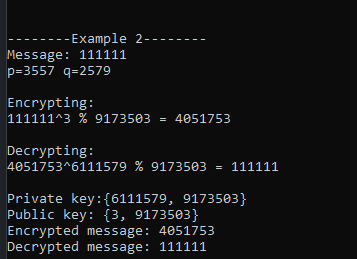
* Потрібно зробити операцію, дуже схожу на вашу, але замість e використовую d. Зводжу E в ступінь d

# Контрольний приклад 1



На скріншоті продемонстровано роботу алгоритму: а саме всі проміжні обчислення, показано результат шифрування та продемонстровано успішне розшифровування.

# Контрольний приклад 2



На скріншоті продемонстровано роботу алгоритму: а саме всі проміжні обчислення, показано результат шифрування та продемонстровано успішне розшифровування.

# Шифрування і розшифрування за алгоритмом RSA

*Таблиця 1.*

|  |  |  |
| --- | --- | --- |
| Крок | Опис операції | Результат операції |
| 1 | Обрати два довільних простих числа *p* і *q*  *p ≠ q*; 1 < *p, q* < 100 | **p = 11**  **q = 29** |
| 2 | Обчислити добуток *n = p* ∙ *q*  ***n* має бути унікальним** в [Таблиця RSA 3 курс](https://docs.google.com/spreadsheets/d/1gtZC9geHMUL4dCh347UmVJKLqED8NJ-CmLgI2NUZxDc/edit?usp=sharing) | **319** |
| 3 | Обчислити функцію Ейлера *φ*(*n*) *=* (*p – 1*) ∙ (*q – 1*) | **280** |
| 4 | Обрати відкриту експоненту *e* : *1 < e < φ(n),  e –* взаємно просте з *φ(n)* | **3** |
| 5 | Обчислити секретну експоненту *d:  (e∙d) mod φ(n) = 1* | **187** |
| 6 | Зберегти закритий ключ {*d*, *n*} | **{187, 319}** |
| 7 | Опублікувати відкритий ключ {*е*, *n*} | **{3, 319}** |
|  |  |  |
| 8 | Одержати від відправника або викладача зашифроване повідомлення *C*. Дії відправника:  1) обрати текст для шифрування *M;* 2) *ci = (mi)e mod n .* | **1)M = ‘sikorsky’**  **2)** **[202, 293, 83, 78, 108, 202, 83, 154]** |
| 9 | Розшифрувати задане повідомлення *C:*  *mi = (ci)d mod n* | **‘sikorsky’** |
| 10 | Відповідь: | **[202, 293, 83, 78, 108, 202, 83, 154] -> ‘sikorsky’** |

# Список літератури

1. Тарнавський Ю.А. Технології захисту інформації [Електронний ресурс] / Ю. А. Тарнавський. – Київ: КПІ ім. Ігоря Сікорського, 2018. – 162 с.
2. Шнайер Б. Прикладная криптография: Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер. – М.: Диалектика, 2003. – 610 с.
3. Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии. – М.: Гелиос АРВ, 2001. – 480 с.
4. Столлингс В. Криптография и защита сетей: принципы и практика, 2-е изд.: Пер. с англ. – М.: «Вильямс», 2001. – 672 с.

# Додаток 1

main.py

from tasks import \*

if \_\_name\_\_ == '\_\_main\_\_':

print('RSA crypting\n')

# example1()

# example2() # it counts around 83 secs, be ready

print('Encrypted message: ', my\_encrypt(message='sikorsky',

info=True),'\n\n')

print('Decrypted message: ', my\_decrypt(crypt\_ascii\_mess = [202, 293, 83, 78, 108, 202, 83, 154],

d=187, n=319,

info=True), '\n\n')

rsa.py

def euler(\_p, \_q):

return (\_p - 1)\*(\_q - 1)

def get\_keys(p, q):

eul = euler(p, q)

n = p \* q

for i in range(3, eul):

if i>1:

for j in range(2,i):

if(i % j==0):

break

else:

if eul % i != 0:

e = i

break

d = 1

while True:

if (d\*e)%eul == 1:

break

else:

d += 1

return n, e, d

def encrypt(mes, e, n, info=False):

ans = mes\*\*e % n

if info:

print('Encrypting:')

print(f'{mes}^{e} % {n} = {ans}\n')

return ans

def decrypt(mes, d, n, info=False):

ans = mes\*\*d % n

if info:

print('Decrypting:')

print(f'{mes}^{d} % {n} = {ans}\n')

return ans

if \_\_name\_\_ == '\_\_main\_\_':

message = 105

p = 11

q = 29

n, e, d = get\_keys(p, q)

enc = encrypt(message, e, n)

dec = decrypt(enc, d, n)

print(f'Message: {message}\np={p} q={q} e={e}\n')

print(f'n={n} phi={euler(p, q)} e={e} d={d}')

print(f'Private key:{ {d, n}}\nPublic key: { {e, n}}')

print(f'Encrypted message: {enc}')

print(f'Decrypted message: {dec}')

tasks.py

from rsa import encrypt, decrypt, get\_keys, euler

def example1():

print('--------Example 1--------')

message = 72

p = 11

q = 17

print(f'Message: {message}\np={p} q={q}\n')

n, e, d = get\_keys(p, q)

enc = encrypt(message, e, n, 1)

dec = decrypt(enc, d, n, 1)

print(f'Private key:{ {d, n}}\nPublic key: { {e, n}}')

print(f'Encrypted message: {enc}')

print(f'Decrypted message: {dec}\n\n')

def example2():

print('--------Example 2--------')

message = 111111

p = 3557

q = 2579

print(f'Message: {message}\np={p} q={q}\n')

n, e, d = get\_keys(p, q)

enc = encrypt(message, e, n, 1)

dec = decrypt(enc, d, n, 1)

print(f'Private key:{ {d, n}}\nPublic key: { {e, n}}')

print(f'Encrypted message: {enc}')

print(f'Decrypted message: {dec}')

def my\_encrypt(message, info=False):

p = 11

q = 29

ascii\_mess = [ord(x) for x in message]

n, e, d = get\_keys(p, q)

if info:

print('--------My encryption--------')

print(f'Message: {message}\nASCII: {ascii\_mess}\np={p} q={q}\n')

print(f'n={n} phi={euler(p, q)} e={e} d={d}')

print(f'Private key:{ {d, n}}\nPublic key: { {e, n}}\n')

enc = []

for letter in ascii\_mess:

enc.append(encrypt(letter, e, n))

if info: print(f'Encrypted letter: {enc[-1]}')

return enc

def my\_decrypt(crypt\_ascii\_mess, d, n, info=False):

if info:

print('--------My decryption--------')

print(f'Encrypted ASCII message: {crypt\_ascii\_mess}\n')

dec = []

for letter in crypt\_ascii\_mess:

dec.append(decrypt(letter, d, n))

if info: print(f'Decrypted letter: {dec[-1]}')

res = ''.join([chr(c) for c in dec])

if info: print(f'\nDecrypted ASCII message: {dec}')

return res

# Додаток 2

